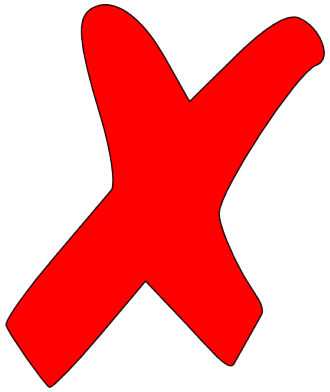

LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891
SPACE RANGER
LIGHTYEAR

# Graphics

Programming takes practice.

Some common issues.

# Reusing Variables

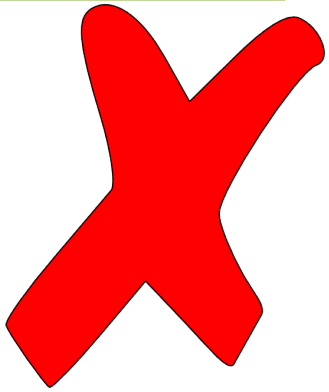


```
16    public void run() {
17        int counter = 0;
18        for (int i = 0; i < 10; i++) {
19            int counter = counter + 1;
20        }
21
22    }
```

You only need to tell Java the type of a variable once.

```
16    public void run() {
17        int counter = 0;
18        for (int i = 0; i < 10; i++) {
19            counter = counter + 1;
20        }
21
22    }
23
```

# Where semicolons; belong

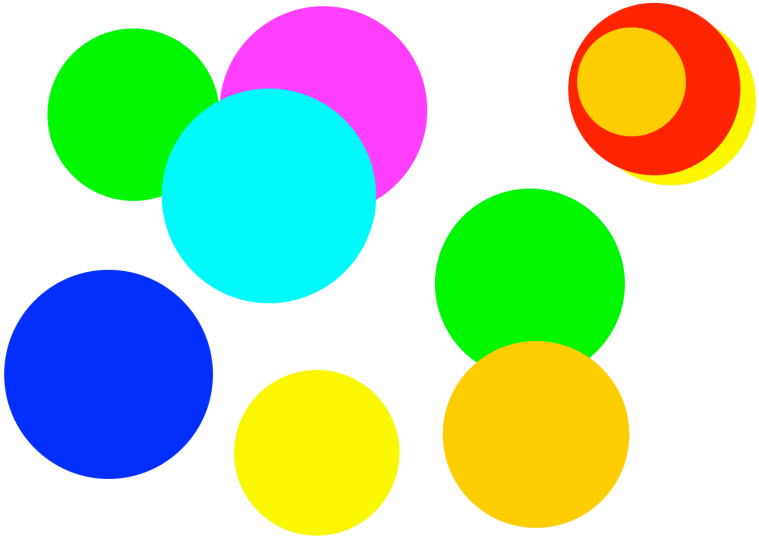


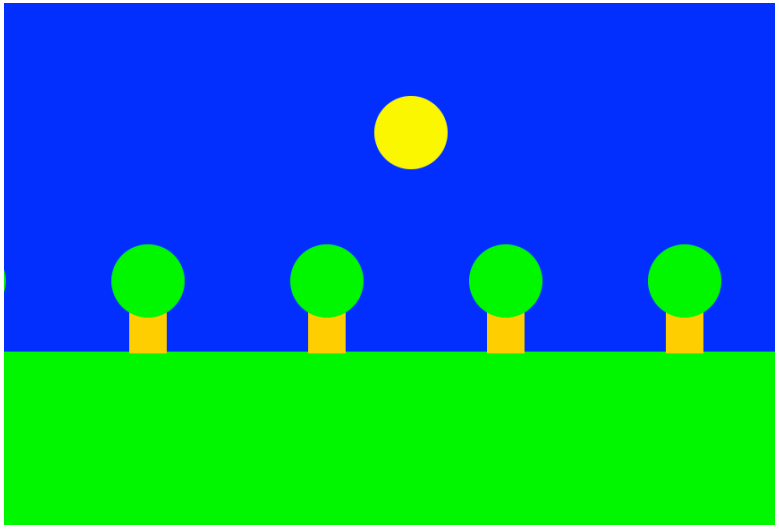
```
public void run() {
    int counter = 0;
    for (int i = 0; i < 10; i++); {
        counter = counter + 1;
    }

}
```

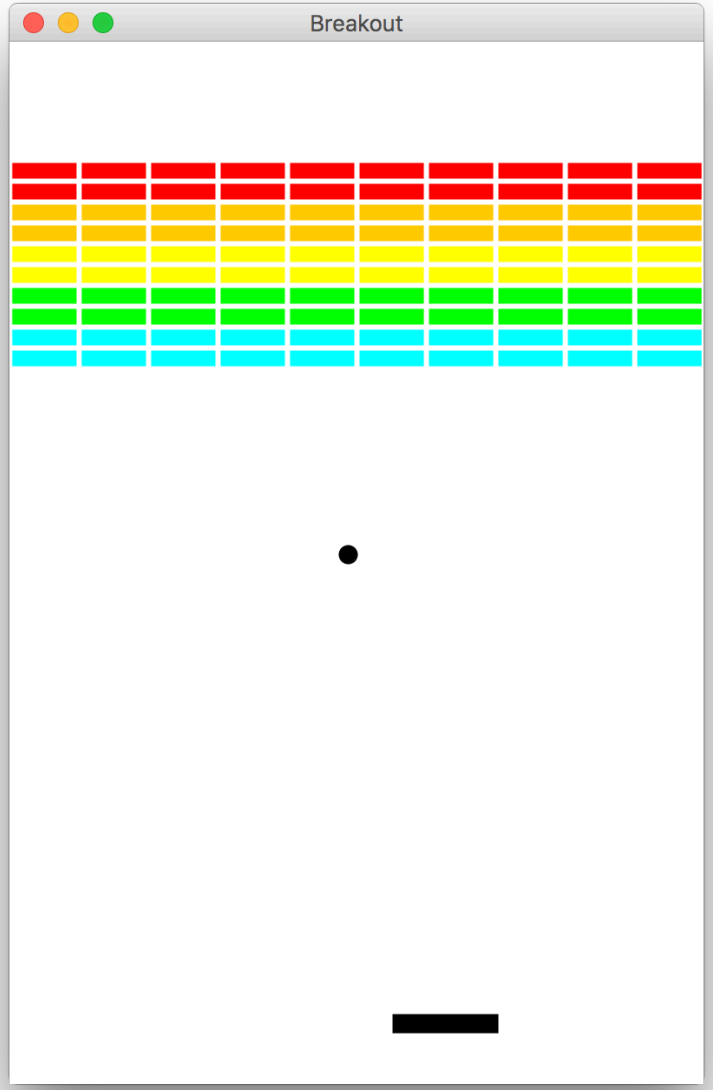
Semicolons do not go after for, while, if, else. **; and { do NOT go together**

```
public void run() {
    int counter = 0;
    for (int i = 0; i < 10; i++) {
        counter = counter + 1;
    }

}
```

# Beyond Console Programs





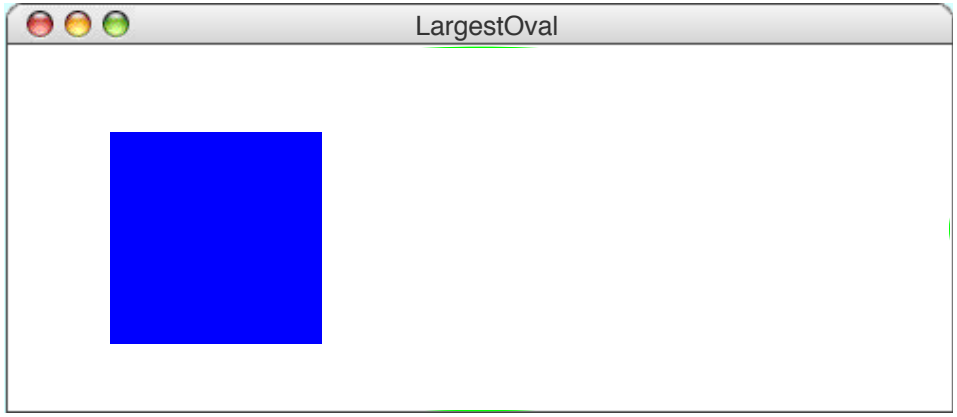


# GRect

GRect is a variable that stores a rectangle.

As an example, the following `run` method displays a rectangle

```
public void run() {
    GRect rect = new GRect(200, 200);
    rect.setFilled(true);
    rect.setColor(Color.BLUE);
    add(rect, 50, 50);
}
```



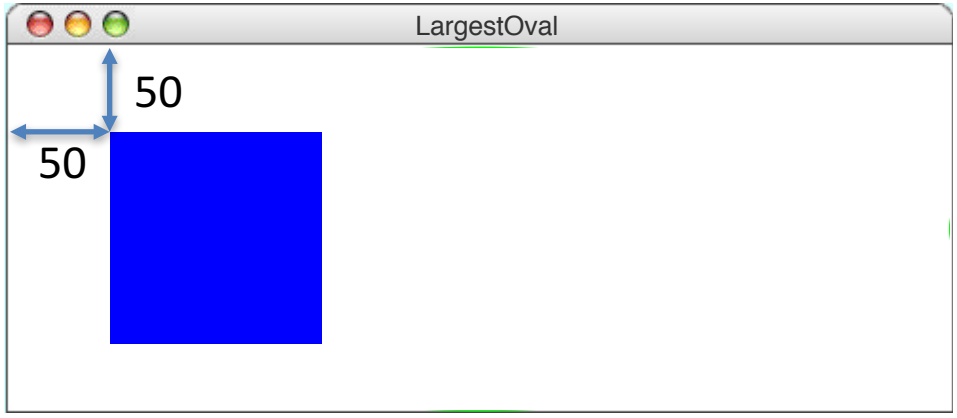
# GRect

GRect is a variable that stores a rectangle.

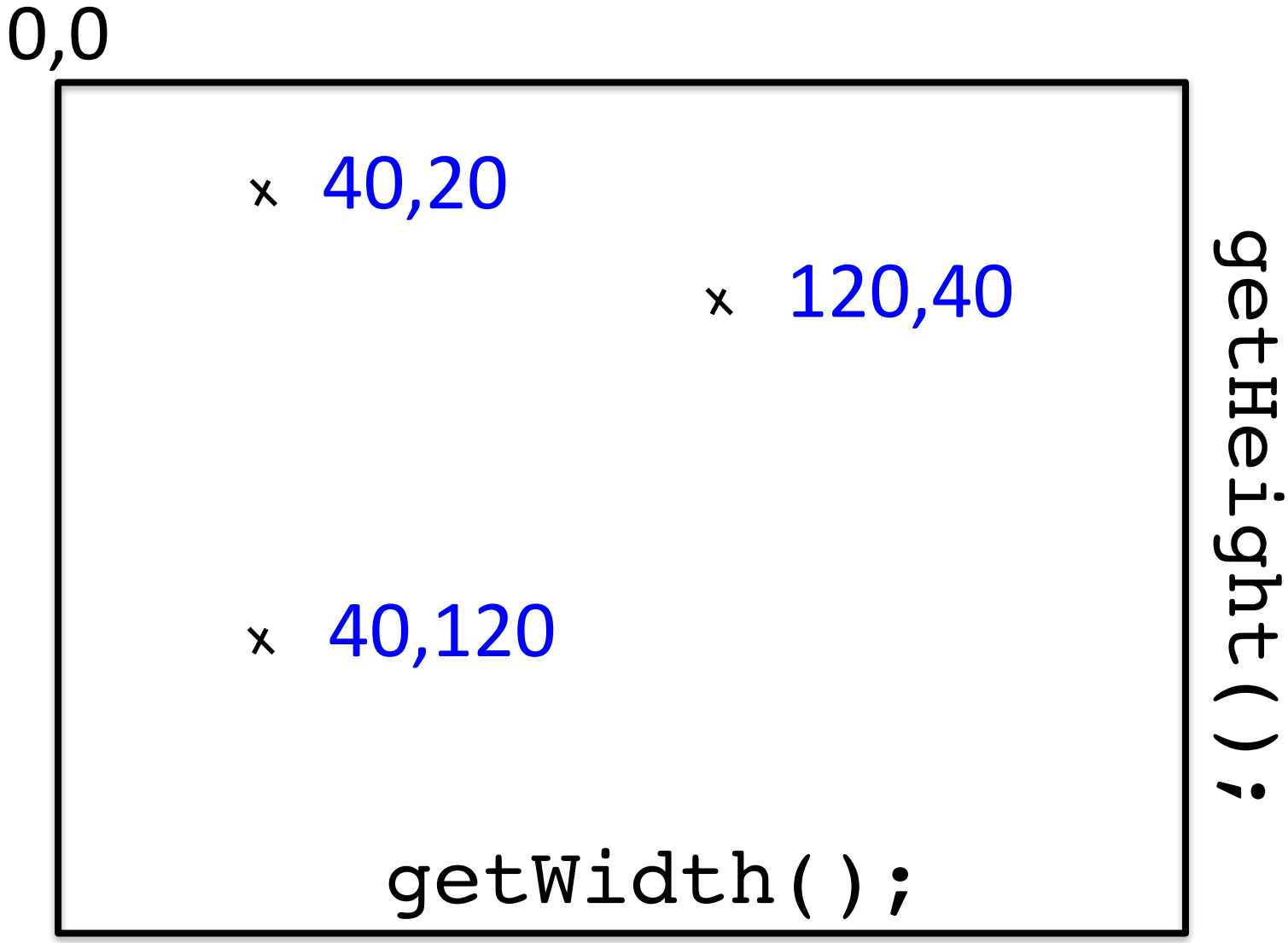
As an example, the following `run` method displays a rectangle

```
public void run() {
    GRect rect = new GRect(200, 200);
    rect.setFilled(true);
    rect.setColor(Color.BLUE);
    add(rect, 50, 50);
}
```

Coordinates for a rectangle are the top left corner.

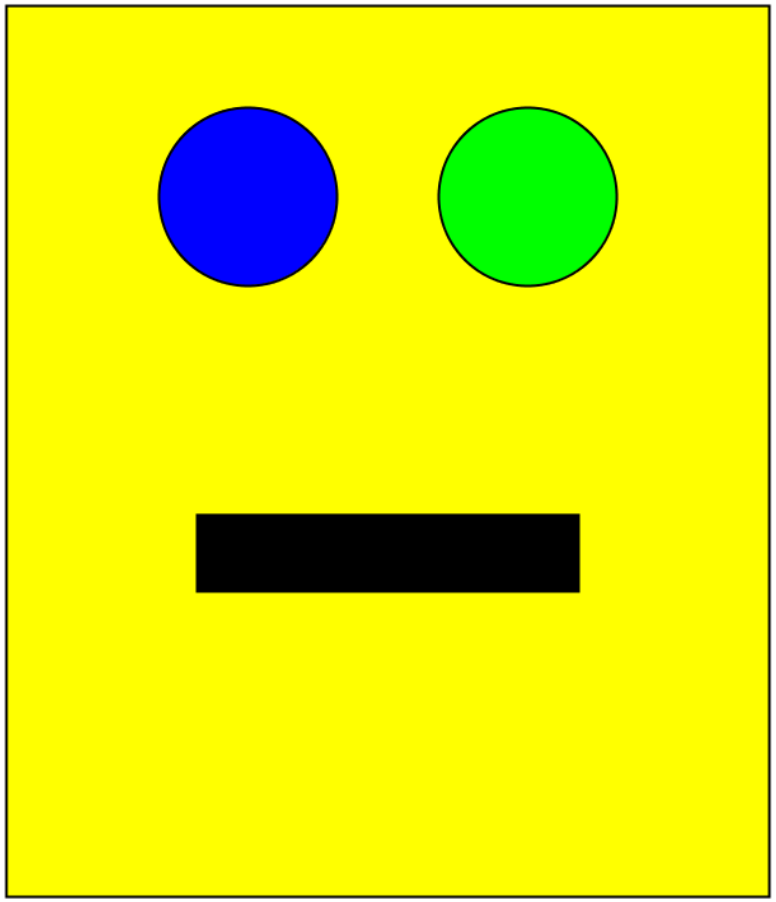


# Graphics Coordinates

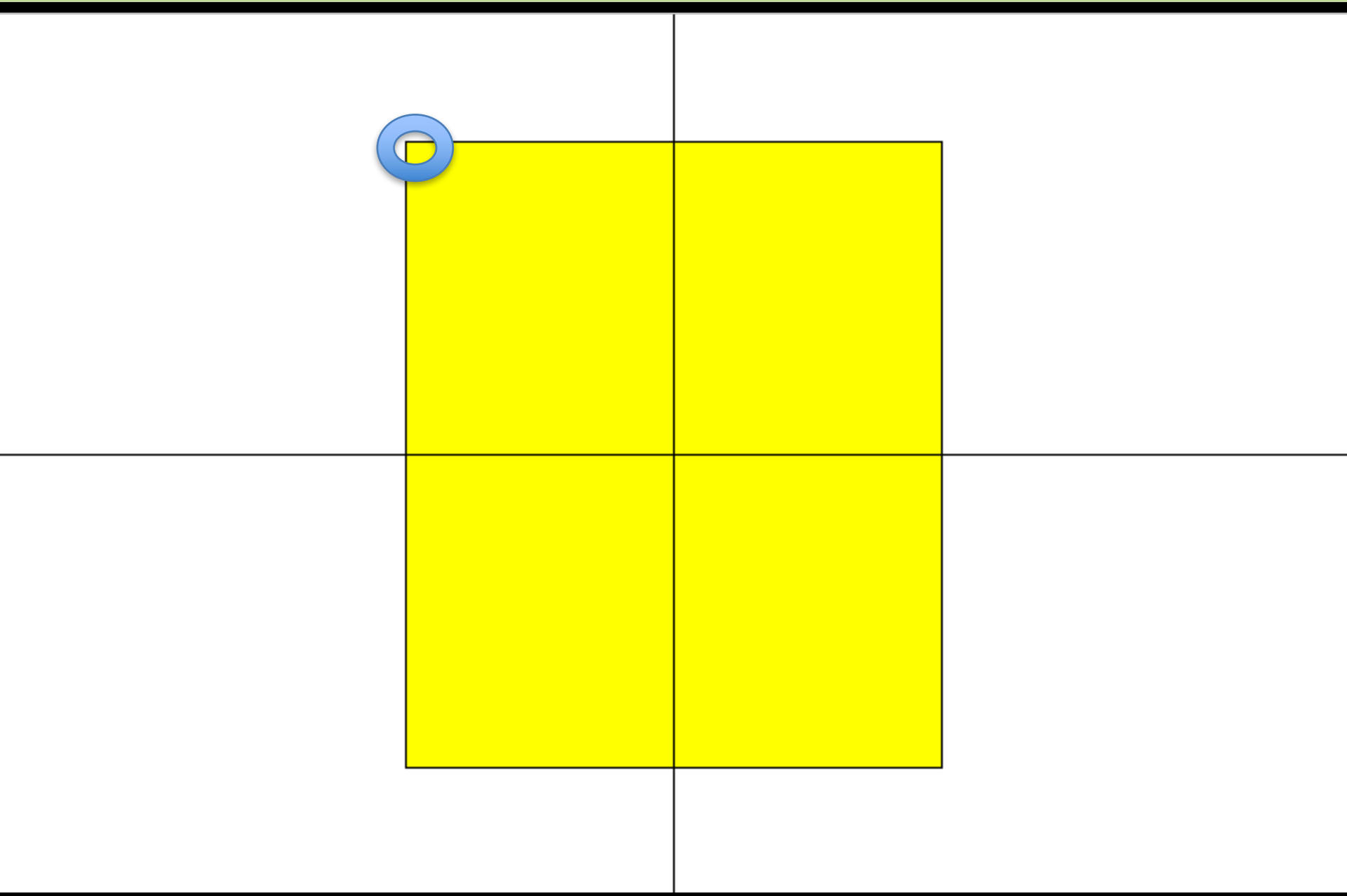


# Karel's Grandpa

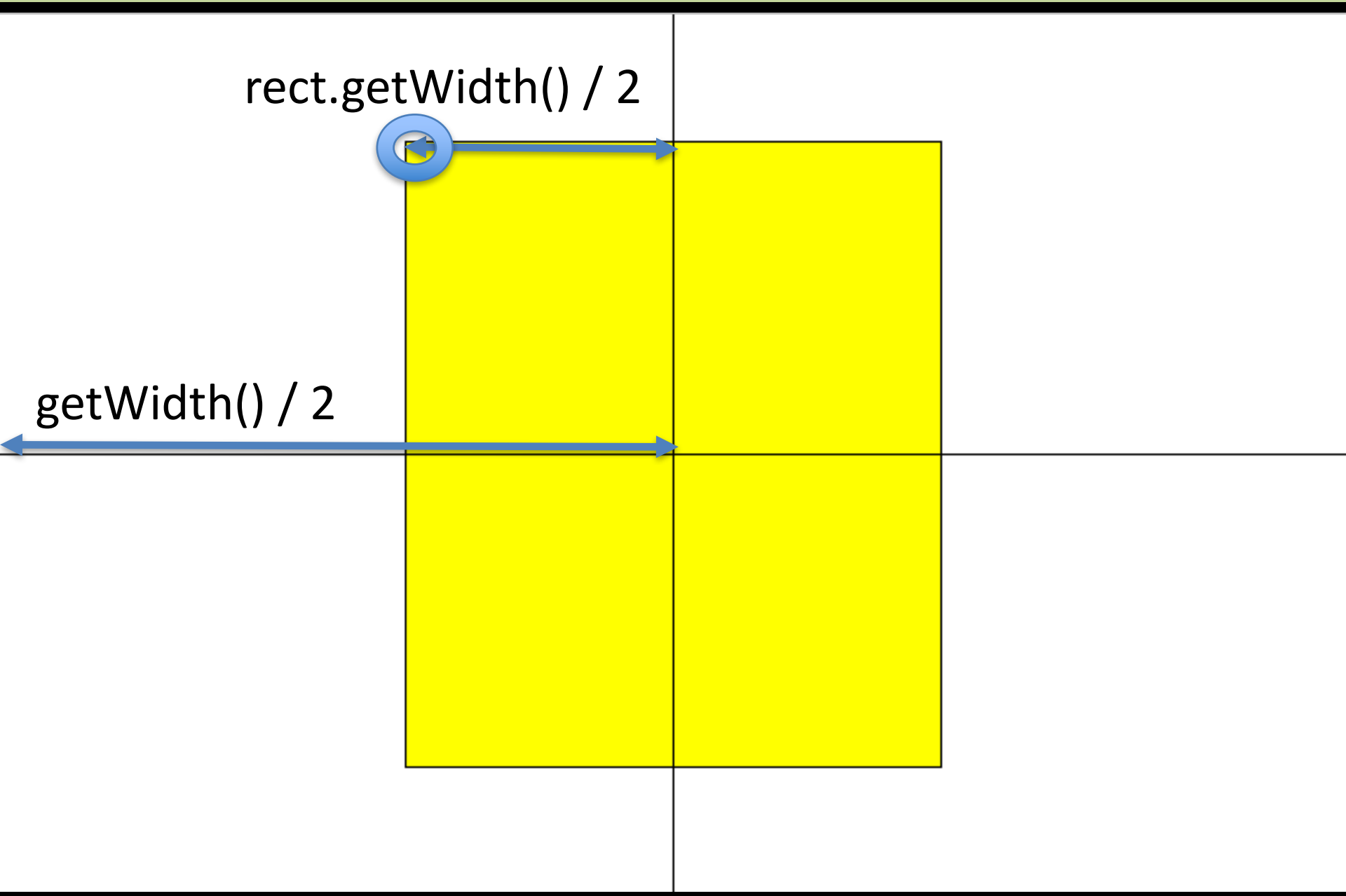
Robot Face



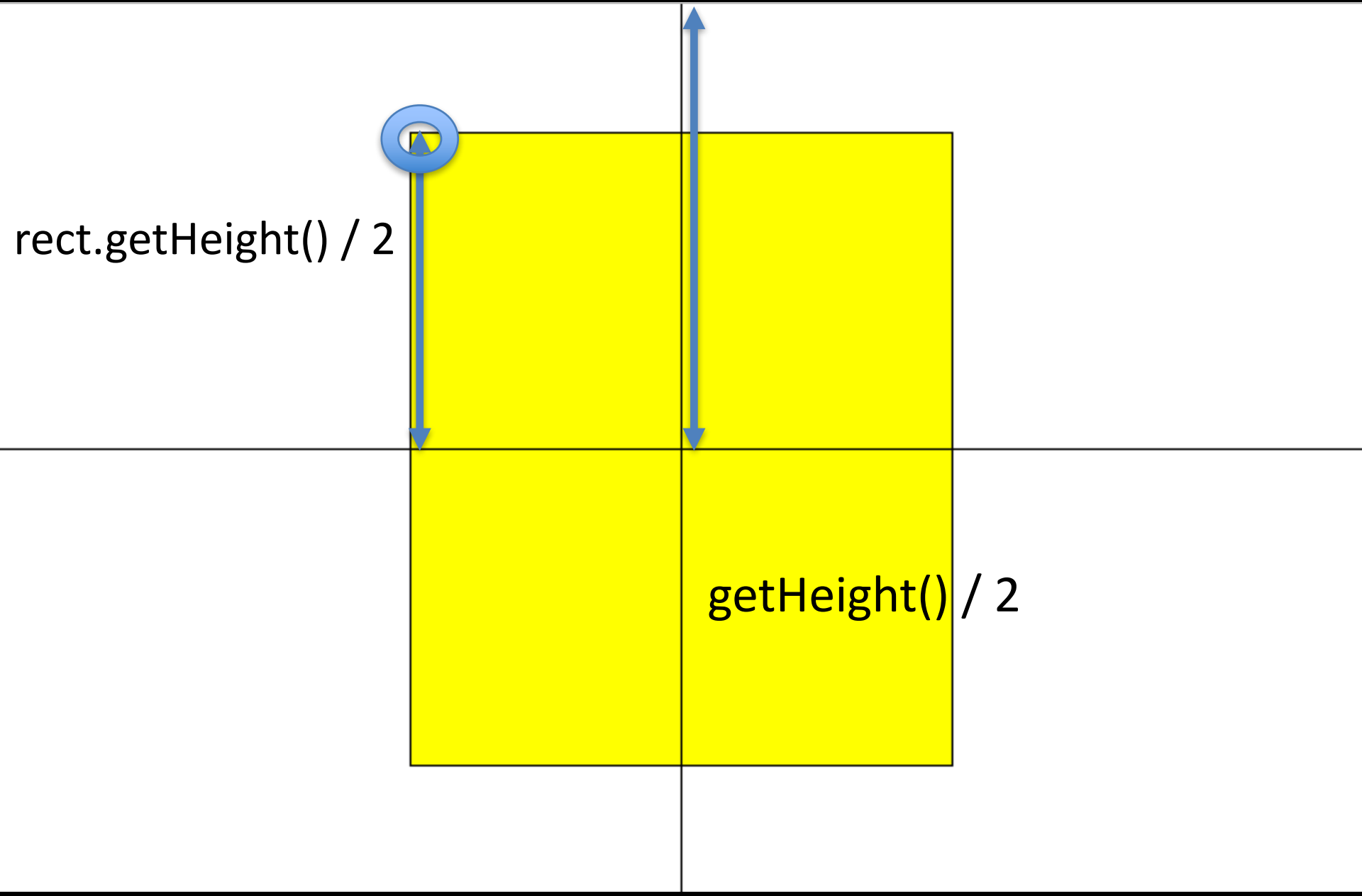
# Karel's Grandpa



# Karel's Grandpa



# Karel's Grandpa

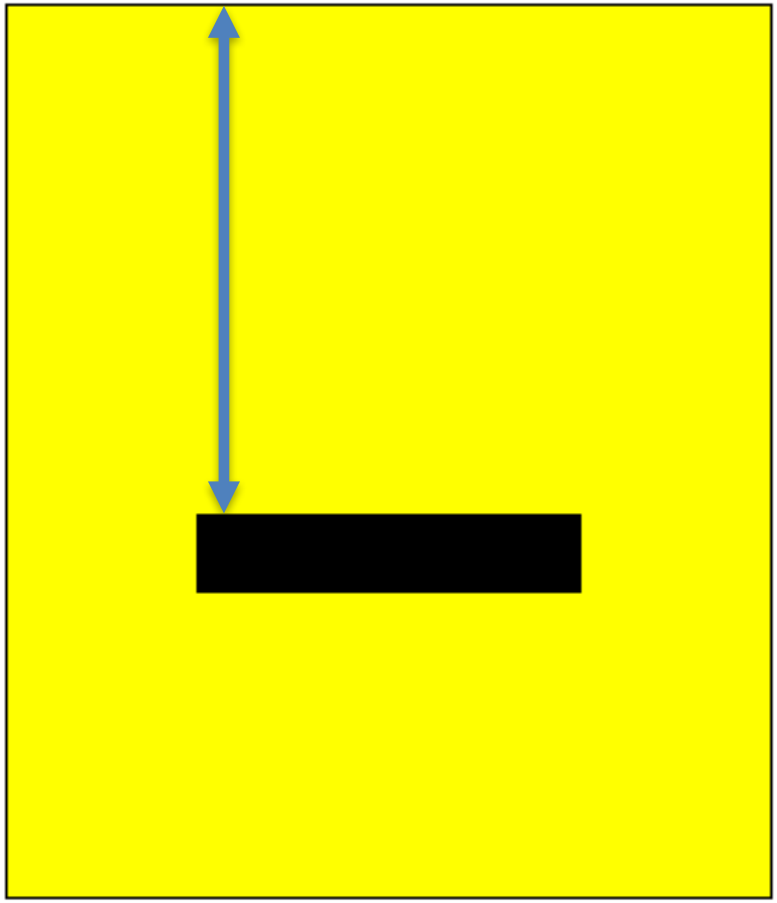


# Constants

```
/*The distance from the top of the head to the top of the mouth:*/
private static final int MOUTH_Y_OFFSET = 200;
```

# Karel's Grandpa

```
/*The distance from the top of the head to the top of the mouth:*/
private static final int MOUTH_Y_OFFSET = 200;
```



# GLabel

A variable that represents text.

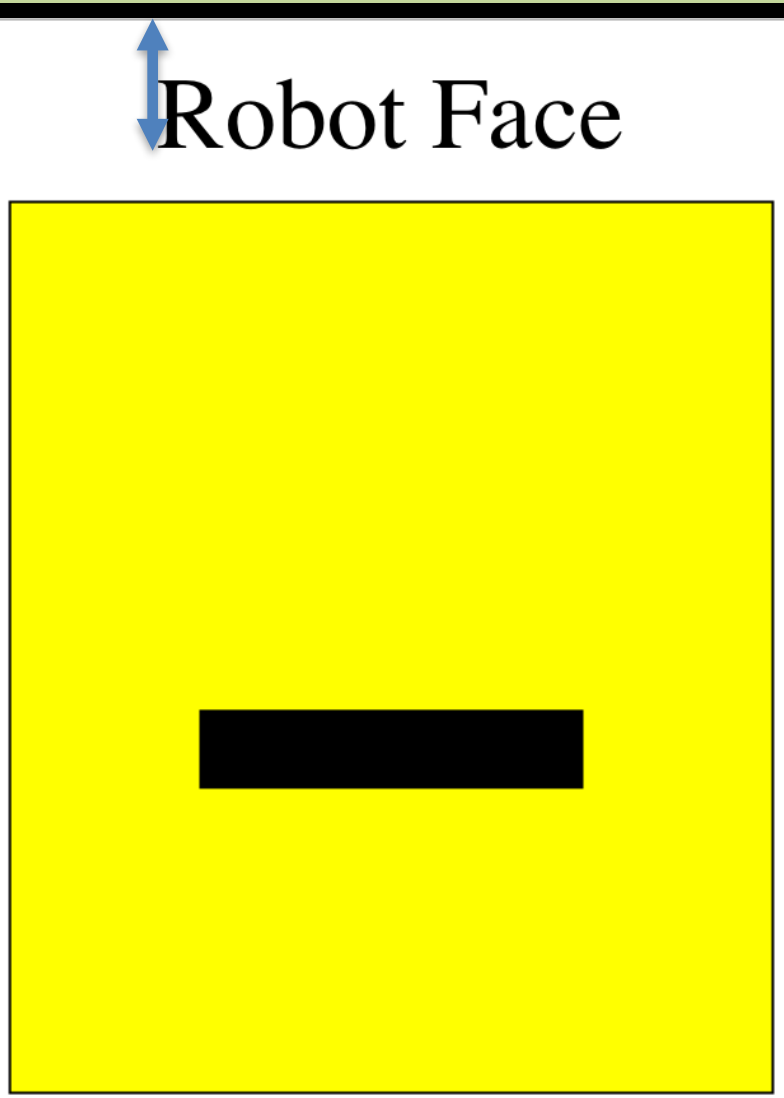
```
public class HelloProgram extends GraphicsProgram {
    public void run() {
        GLabel label = new GLabel("hello, world", 100, 75);
        label.setFont("SansSerif-36");
        label.setColor(Color.RED);
        add(label);
    }
}
```

# GLabel

A variable that represents text.

```
public class HelloProgram extends GraphicsProgram {
    public void run() {
        GLabel label = new GLabel("hello, world", 100, 75);
        label.setFont("SansSerif-36");
        label.setColor(Color.RED);
        add(label);
    }
}
```

# Karel's Grandpa



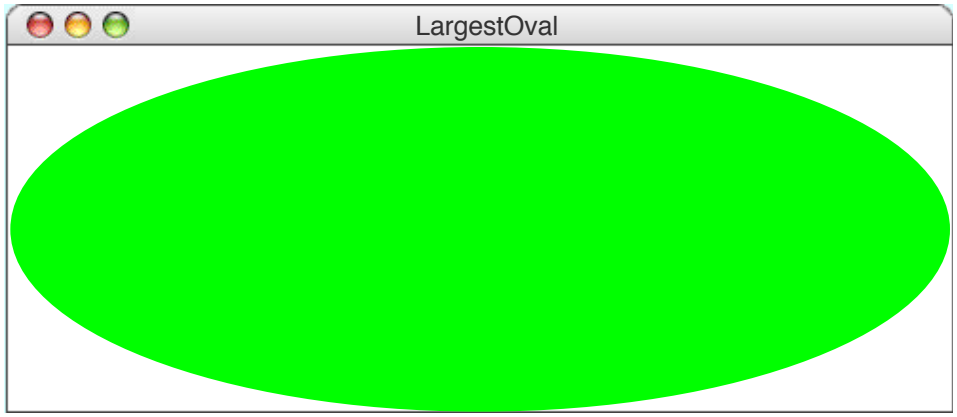

```
/*The distance from the top of the screen to the base of the label:*/
private static final int LABEL_Y = 50;
```

# GOval

The `GOval` class represents an elliptical shape defined by the boundaries of its enclosing rectangle.

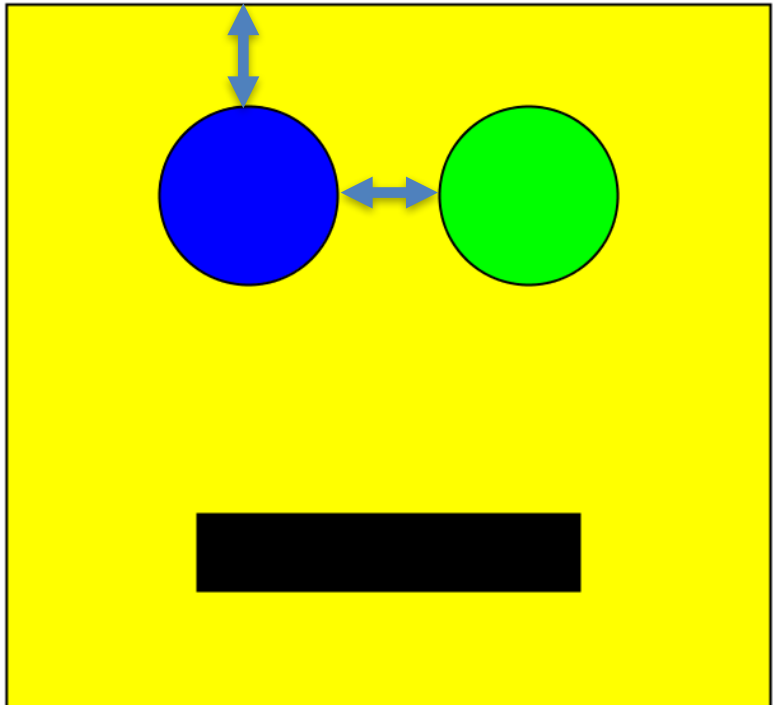
As an example, the following `run` method creates the largest oval that fits within the canvas:

```
public void run() {
    GOval oval = new GOval(getWidth(), getHeight());
    oval.setFilled(true);
    oval.setColor(Color.GREEN);
    add(oval, 0, 0);
}
```



# Karel's Grandpa

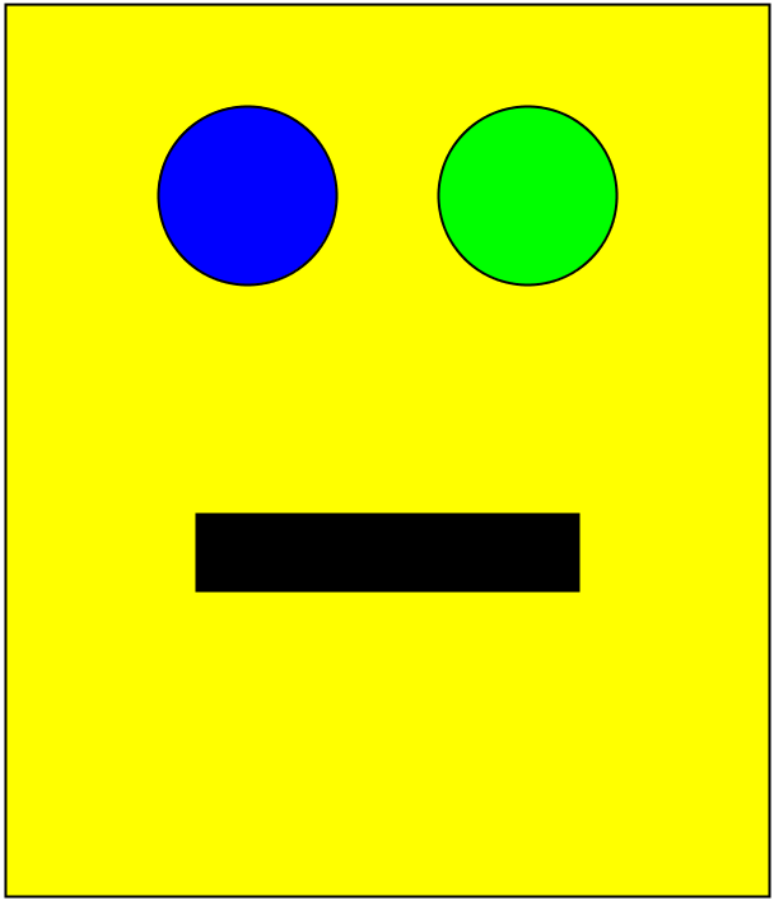
## Robot Face



```
/*The distance from the top of the head to the top of the eyes:*/
private static final int EYE_Y_OFFSET = 40;

/*The distance in between the two eyes:*/
private static final int EYE_X_SEPARATION = 40;
```

# Karel's Grandpa

Robot Face



# Graphics Methods

| | |
|---|---|
| `add(`*object*`)` | Adds the object to the canvas at the front of the stack |
| `add(`*object*`, x, y)` | Moves the object to (*x*, *y*) and then adds it to the canvas |
| `remove(`*object*`)` | Removes the object from the canvas |
| `removeAll()` | Removes all objects from the canvas |
| `getElementAt(`*x*`, `*y*`)` | Returns the frontmost object at (*x*, *y*), or `null` if none |
| `getWidth()` | Returns the width in pixels of the entire canvas |
| `getHeight()` | Returns the height in pixels of the entire canvas |
| `setBackground(`*c*`)` | Sets the background color of the canvas to *c.* |
| `pause(`*milliseconds*`)` | Pauses the program for the specified time in milliseconds |
| `waitForClick()` | Suspends the program until the user clicks the mouse |

# Graphics Methods

| | |
|---|---|
| `add(`*object*`)` | Adds the object to the canvas at the front of the stack |
| `add(`*object*`, x, y)` | Moves the object to (*x*, *y*) and then adds it to the canvas |
| `remove(`*object*`)` | Removes the object from the canvas |
| `removeAll()` | Removes all objects from the canvas |
| `getElementAt(x, y)` | Returns the frontmost object at (*x*, *y*), or `null` if none |
| `getWidth()` | Returns the width in pixels of the entire canvas |
| `getHeight()` | Returns the height in pixels of the entire canvas |
| `setBackground(c)` | Sets the background color of the canvas to *c.* |
| `pause(`*milliseconds*`)` | Pauses the program for the specified time in milliseconds |
| `waitForClick()` | Suspends the program until the user clicks the mouse |

# Reference Sheet

## Constructors

**`new GLabel(String text)`** *or* **`new GLabel(String text, double x, double y)`**
Creates a new **`GLabel`** object; the second form sets its location as well.

**`new GRect(double x, double y, double width, double height)`**
Creates a new **`GRect`** object; the **`x`** and **`y`** parameters can be omitted and default to 0.

**`new GOval(double x, double y, double width, double height)`**
Creates a new **`GOval`** object; the **`x`** and **`y`** parameters can be omitted and default to 0.

**`new GLine(double x1, double y1, double x2, double y2)`**
Creates a new **`GLine`** object connecting (**`x1`**, **`y1`**) and (**`x2`**, **`y2`**).

## Methods common to all graphical objects

**`void setLocation(double x, double y)`**
Sets the location of this object to the specified coordinates.

**`void move(double dx, double dy)`**
Moves the object using the displacements **`dx`** and **`dy`**.

**`double getWidth()`**
Returns the width of the object.

**`double getHeight()`**
Returns the height of the object.

**`void setColor(Color c)`**
Sets the color of the object.

## Methods available for GRect and GOval only

**`void setFilled(boolean fill)`**
Sets whether this object is filled (**`true`** means filled, **`false`** means outlined).

**`boolean isFilled()`**
Returns **`true`** if the object is filled.

**`void setFillColor(Color c)`**
Sets the color used to fill this object. If the color is **`null`**, filling uses the color of the object.

## Methods available for GLabel only

**`void setFont(String fontName)`**
Sets the font, as described in Chapter 5.

**`double getAscent()`**
Returns the height above the text baseline.

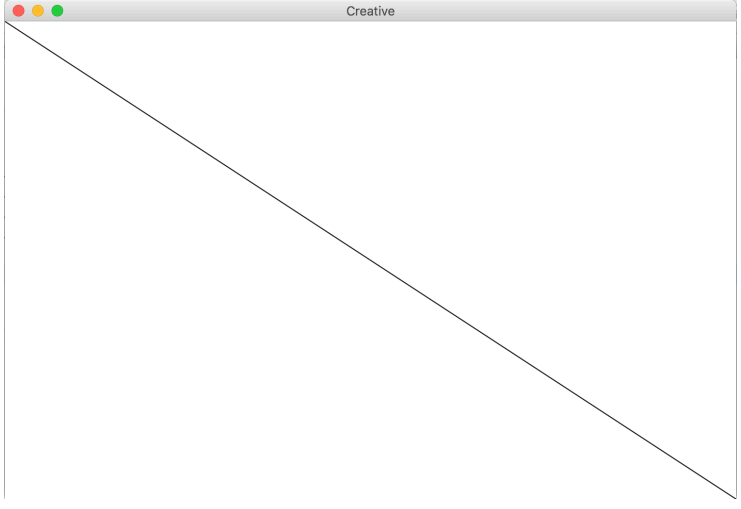
**`double getDescent()`**
Returns the height below the text baseline.

# GLine

The `GLine` class represents a line defined by a start point and an end point.

As an example, the following `run` method creates a diagonal line across the canvas:

```
public void run() {
    GLine line = new GLine(0,0, getWidth(), getHeight());
    add(line);
}
```



# Karel's Grandpa